

CS 1124

Media Computation

Steve Harrison

Lecture 4.2 (September 18, 2008)

Today ...

- Moving to chapter 5 ...
- HW 4
- Group project

Today


- How can we merge pictures?
- Blending pictures together
 - **blend 1 mix two pictures together**
 - **blend 2 (from the book) overlap two pictures**
 - **blend 3 (iTunes) mirror effect**
- Homework 4

Merging a picture at the pixel-level

- What happens if we want to combine pictures?
- We need to combine pixels, right?
- Some options:
 - **replace pixels (doing that in HW 3)**
 - **use logical combination of some sort....**
 - **add the colors together**


Adding colors together in a pixel

■ Blue + Red



The diagram illustrates the addition of two pixels. On the left, a blue square contains the text "0, 0, 127". To its right is a plus sign, followed by a red square containing "127, 0, 0". To the right of that is an equals sign, followed by a purple square containing "127, 0, 127".

■ Blue + Blue



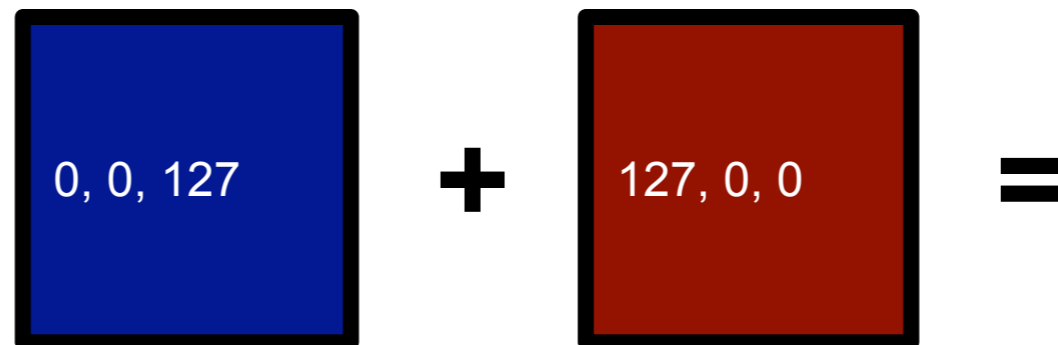
The diagram illustrates the addition of two pixels. On the left, a blue square contains the text "0, 0, 127". To its right is a plus sign, followed by another blue square containing "0, 0, 127". To the right of that is an equals sign, followed by a darker blue square containing "0, 0, 254".

- That doesn't look right does it? If we have two pictures with the same color, then the new picture should be that color, right?

Adding colors together in a pixel

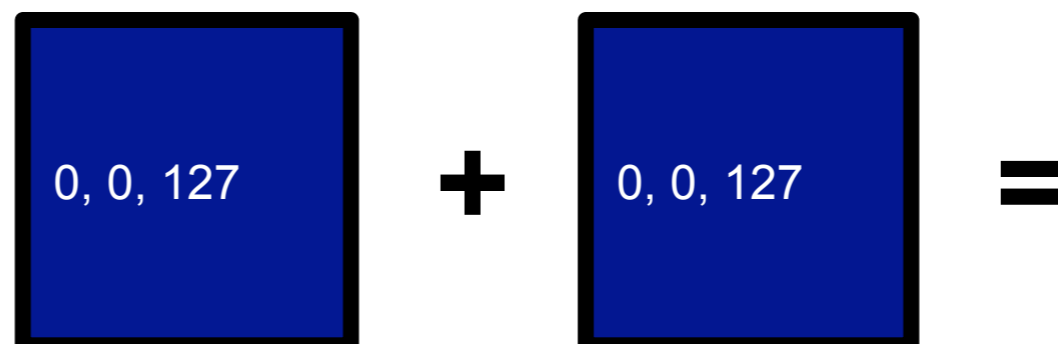
- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red



A diagram illustrating the addition of two colors. On the left is a blue square with the text "0, 0, 127". To its right is a plus sign "+". To the right of the plus sign is a red square with the text "127, 0, 0". To the right of the red square is an equals sign "=".

- Blue + Blue



A diagram illustrating the addition of two identical colors. On the left is a blue square with the text "0, 0, 127". To its right is a plus sign "+". To the right of the plus sign is another blue square with the text "0, 0, 127". To the right of the second blue square is an equals sign "=".

Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red

A diagram illustrating the addition of two colors. On the left is a blue square with the text "50%" and "0, 0, 127". To its right is a plus sign, followed by a red square with the text "127, 0, 0". To the right of the red square is an equals sign.

- Blue + Blue

A diagram illustrating the addition of two colors. On the left is a blue square with the text "0, 0, 127". To its right is a plus sign, followed by another blue square with the text "0, 0, 127". To the right of the second square is an equals sign.

Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red

A diagram illustrating the addition of two colors. On the left is a blue square with the text "50%" and "0, 0, 127". To its right is a plus sign, followed by a red square with the text "50%" and "127, 0, 0". To the right of the red square is an equals sign.

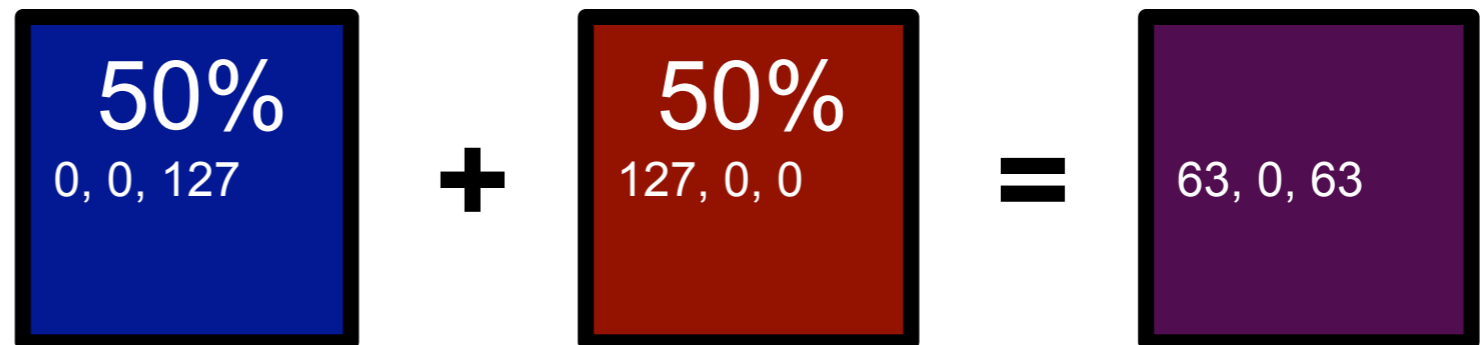
- Blue + Blue

A diagram illustrating the addition of two colors. On the left is a blue square with the text "0, 0, 127". To its right is a plus sign, followed by another blue square with the text "0, 0, 127". To the right of the second square is an equals sign.

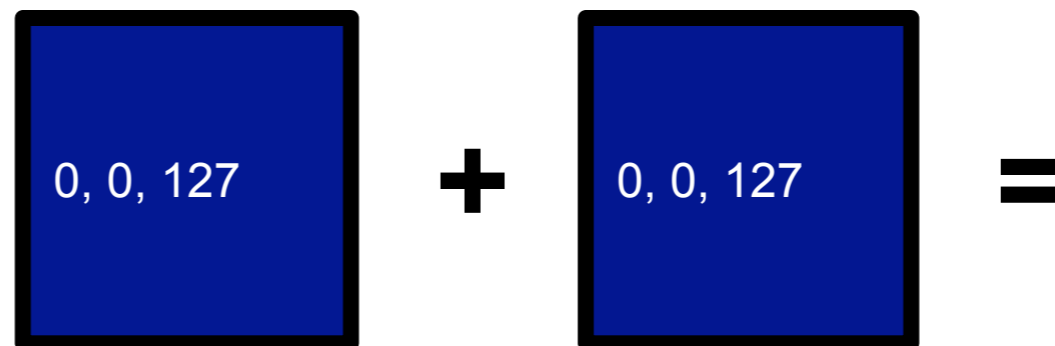
Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red



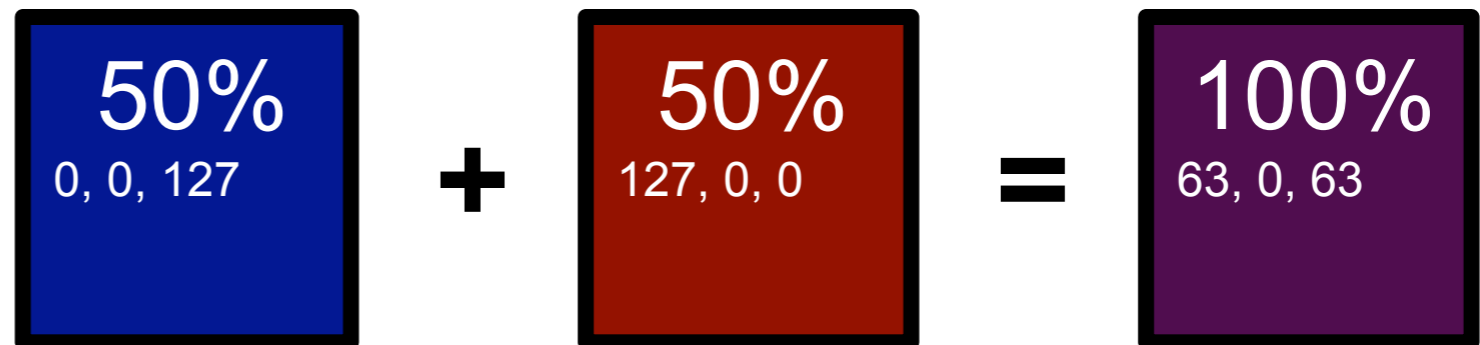
- Blue + Blue



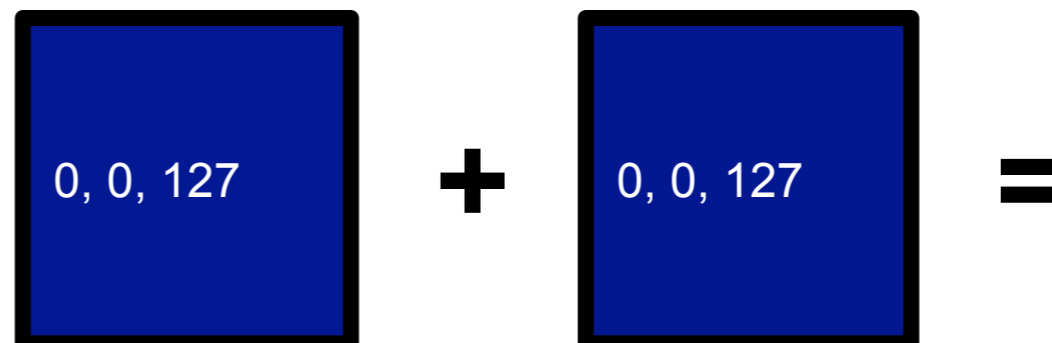
Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red



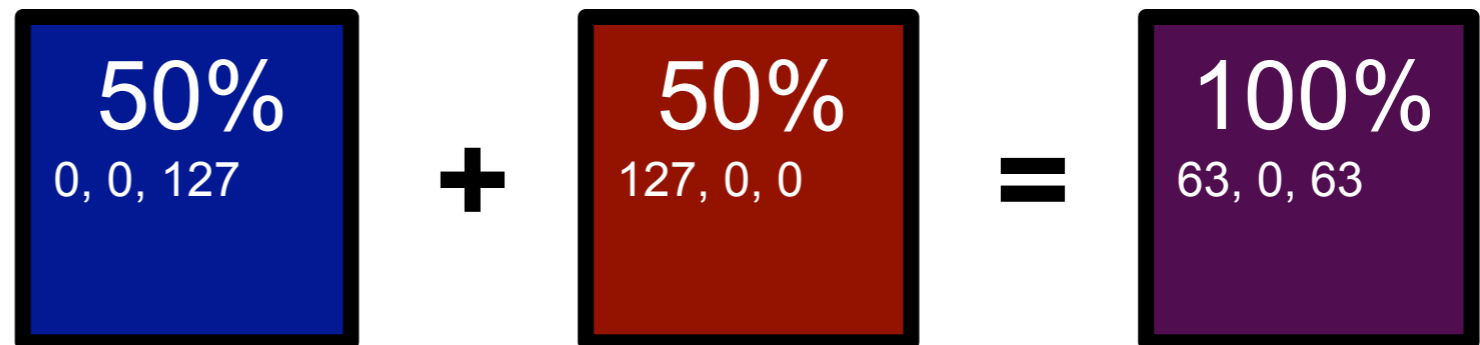
- Blue + Blue



Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red



- Blue + Blue



Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red

A diagram illustrating the addition of two colors. On the left is a blue square labeled "50%" with the RGB values "0, 0, 127". To its right is a plus sign, followed by a red square labeled "50%" with the RGB values "127, 0, 0". To the right of the red square is an equals sign, followed by a purple square labeled "100%" with the RGB values "63, 0, 63".

- Blue + Blue

A diagram illustrating the addition of two colors. On the left is a blue square labeled "50%" with the RGB values "0, 0, 127". To its right is a plus sign, followed by another blue square labeled "50%" with the RGB values "0, 0, 127". To the right of the second blue square is an equals sign, followed by a darker blue square with the RGB values "126, 0, 0".

Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red

A diagram illustrating the addition of two colors. On the left, a blue square contains the text "50%" and "0, 0, 127". To its right is a plus sign. Next is a red square containing "50%" and "127, 0, 0". To its right is an equals sign. Finally, a purple square contains "100%" and "63, 0, 63".

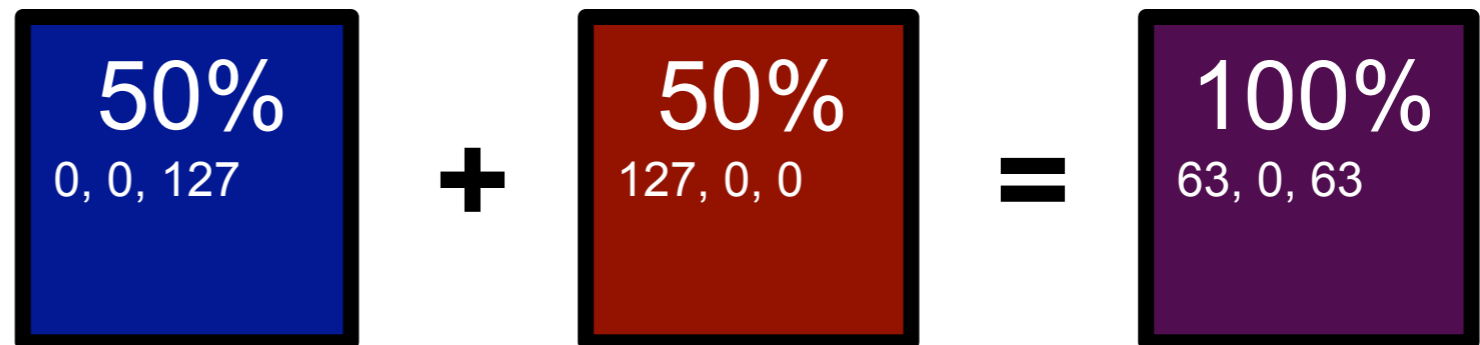
- Blue + Blue

A diagram illustrating the addition of two colors. On the left, a blue square contains the text "50%" and "0, 0, 127". To its right is a plus sign. Next is another blue square containing "50%" and "0, 0, 127". To its right is an equals sign. Finally, a darker blue square contains "126, 0, 0".

Adding colors together in a pixel

- Lets say each pixel contributes only its portion of the color. If the new color should 100% then each color should add up to 100% Merging two colors means 50% each

- Blue + Red



- Blue + Blue



Adding colors together in a pixel






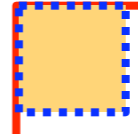


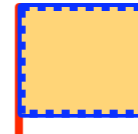


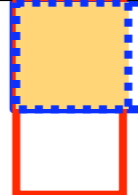
- So what if we had three pictures we were combining?
 - **each would get 33.33%, right?**
- What we wanted one picture to be more visible?
 - **we could increase its percentage and decrease the other pictures' percentages**

Blending pictures together (1)

- 50% of picture + 50% of another = blended image!
 - works on a pixel-by-pixel / color-by-color basis!
 - So picture is overlapping area
- psuedo code
 - a “program” made of comments
 - a template to write the program
- blend 1 (file1, file2)
 - get the pictures in each file
 - make a canvas for blended picture
 - for each pixel add 50% of each color picture1 to 50% of each color of picture2, put into canvas

How big is the resulting picture?

- Why we get the least width and least height
- Consider possible sizes of source1 & source2:

 100 x 100	 100 x 100	 100 x 100
 100 x 100	 80 x 80	 80 x 80
 100 x 100	 200 x 80	 100 x 80
 80 x 150	 200 x 80	 80 x 80



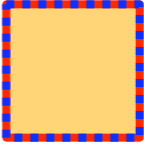


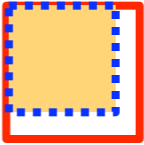
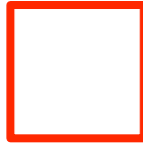

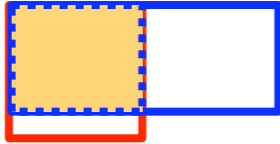


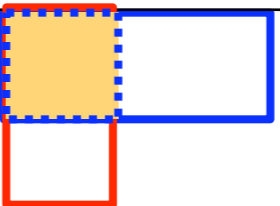
Blending pictures together (1)

```
def blendTwoPictures( fileName1, filename2 ):
    # get the pictures in each file
    source1 = makePicture( fileName1 )
    source2 = makePicture( fileName2 )
    # get the least width and height (Why?)
    canvasX = min( getWidth( source1), getWidth( source2 ) ) + 1
    canvasY = min( getHeight( source1), getHeight( source2 ) ) + 1
    # make a canvas for the blended file
    canvas = makeEmptyPicture( canvasX, canvasY )
    # for each pixel add 50% of each color picture1 to 50% of each color of picture2, put
    # into canvas
    for x in range(1, canvasX ) :
        for y in range( 1, canvasY ) :
            source1Pixel = getPixel( source1, x, y )
            source2Pixel = getPixel( source2, x, y )
            blendRed = (getRed( source1Pixel) * 0.5) + (getRed(source2Pixel) * 0.5)
            blendBlue = (getBlue( source1Pixel) * 0.5) + (getBlue(source2Pixel) * 0.5)
            blendGreen = (getGreen( source1Pixel) * 0.5) + (getGreen(source2Pixel) * 0.5)
            blendColor = makeColor( blendRed, blendGreen, blendBlue )
            setColor( getPixel( canvas, x, y ), blendColor )
    return canvas
```

Bug Alert

Calculating the colors in the order red, blue, green produces the same result, but since we MUST setColor in the order red, green, and blue, it is bad practice

What are some other ways to size the resulting picture?

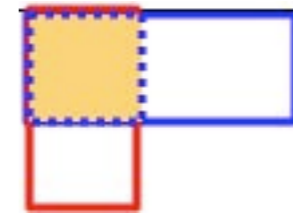
	100 x 100		100 x 100		100 x 100
	100 x 100		80 x 80		80 x 80
	100 x 100		200 x 80		100 x 80
	80 x 150		200 x 80		80 x 80

What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - **need to copy in missing rectangles**
- fit source1 on source2 (or source2 on source1)
 - **scale in x and y to match**
- center source1 over center of source 2
 - **could just take overlap (what size is that?)**
 - **could center + take biggest width & height -->**
 - need to fill in missing top, bottom and sides
- Are there any other ways?

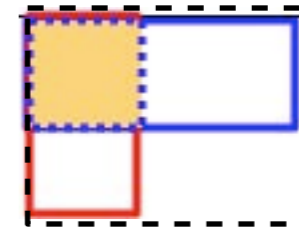
What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



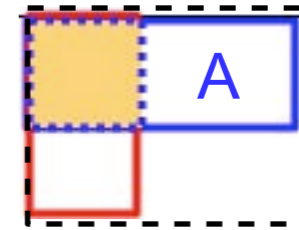
What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



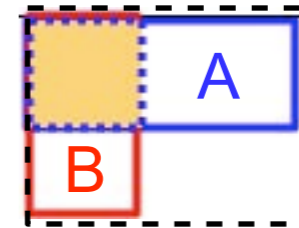
What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



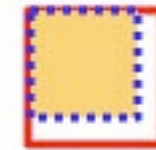
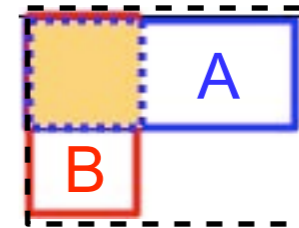
What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



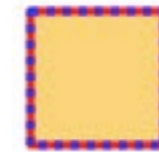
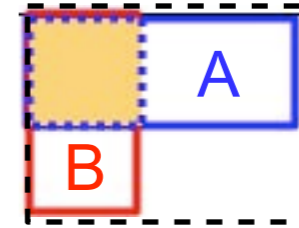
What are some other ways to size the resulting picture?

- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



What are some other ways to size the resulting picture?

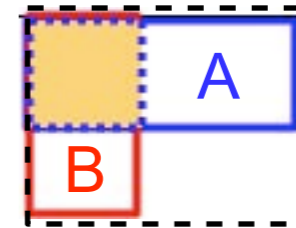
- Biggest width and biggest height
 - need to copy in missing rectangles
- fit source1 on source2 (or source2 on source1)
 - scale in x and y to match
- center source1 over center of source 2
 - could just take overlap (what size is that?)
 - could center + take biggest width & height -->
 - need to fill in missing top, bottom and sides
- Are there any other ways?



What are some other ways to size the resulting picture?

- Biggest width and biggest height

- need to copy in missing rectangles



- fit source1 on source2 (or source2 on source1)

- scale in x and y to match

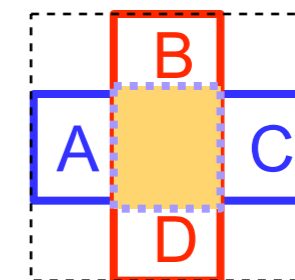


- center source1 over center of source 2

- could just take overlap (what size is that?)

- could center + take biggest width & height -->

- need to fill in missing top, bottom and sides



- Are there any other ways?

Lets blend two blended pictures

```
def blendTwoPictures( fileName1, filename2 ):
    # get the pictures in each file
    source1 = makePicture( fileName1 )
    source2 = makePicture( fileName2 )
    # get the least width and height (Why?)
    canvasX = min( getWidth( source1), getWidth( source2 ) ) + 1
    canvasY = min( getHeight( source1), getHeight( source2 ) ) + 1
    # make a canvas for the blended file
    canvas = makeEmptyPicture( canvasX, canvasY )
    # for each pixel add 50% of each color picture1 to 50% of each color of picture2, put into canvas
    for x in range(1, canvasX ) :
        for y in range( 1, canvasY ) :
            source1Pixel = getPixel( source1, x, y )
            source2Pixel = getPixel( source2, x, y )
            blendRed = (getRed( source1Pixel) * 0.5) + (getRed(source2Pixel) * 0.5)
            blendBlue = (getBlue( source1Pixel) * 0.5) + (getBlue(source2Pixel) * 0.5)
            blendGreen = (getGreen( source1Pixel) * 0.5) + (getGreen(source2Pixel) * 0.5)
            blendColor = makeColor( blendRed, blendBlue, blendGreen )
            setColor( getPixel( canvas, x, y ), blendColor )
    return canvas
```

```
>>> blended = blendTwoPictures( barbFile, katieFile )
>>> barch = blendTwoPictures( file2, file3 )
>>> show(barch)
>>> bbak = blendTwoPictures( barch, blended )
```

The error was:isabs() argument must be a string object, not instance of 'Picture'
Inappropriate argument type.

An attempt was made to call a function with a parameter of an invalid type. This means that you did something such as trying to pass a string to a method that is expecting an integer.

Please check line 3 of /Users/steveharrison/Desktop/2984 Media Computation/class demos/blendTwoPictures

Instead of passing in files, pass in pictures

```
def blendTwoPics( source1, source2 ):
    # get the least width and height (Why?)
    canvasX = min( getWidth( source1), getWidth( source2 ) ) + 1
    canvasY = min( getHeight( source1), getHeight( source2 ) ) + 1
    # make a canvas for the blended file
    canvas = makeEmptyPicture( canvasX, canvasY )
    # for each pixel add 50% of each color picture1 to 50% of each color of picture2, put into canvas
    for x in range(1, canvasX ) :
        for y in range( 1, canvasY ) :
            source1Pixel = getPixel( source1, x, y )
            source2Pixel = getPixel( source2, x, y )
            blendRed = (getRed( source1Pixel) * 0.5) + (getRed(source2Pixel) * 0.5)
            blendBlue = (getBlue( source1Pixel) * 0.5) + (getBlue(source2Pixel) * 0.5)
            blendGreen = (getGreen( source1Pixel) * 0.5) + (getGreen(source2Pixel) * 0.5)
            blendColor = makeColor( blendRed, blendBlue, blendGreen )
            setColor( getPixel( canvas, x, y ), blendColor )
    return canvas
```

```
>>> bbak = blendTwoPics( barch, blended )
>>> show(bbak)
```


Today

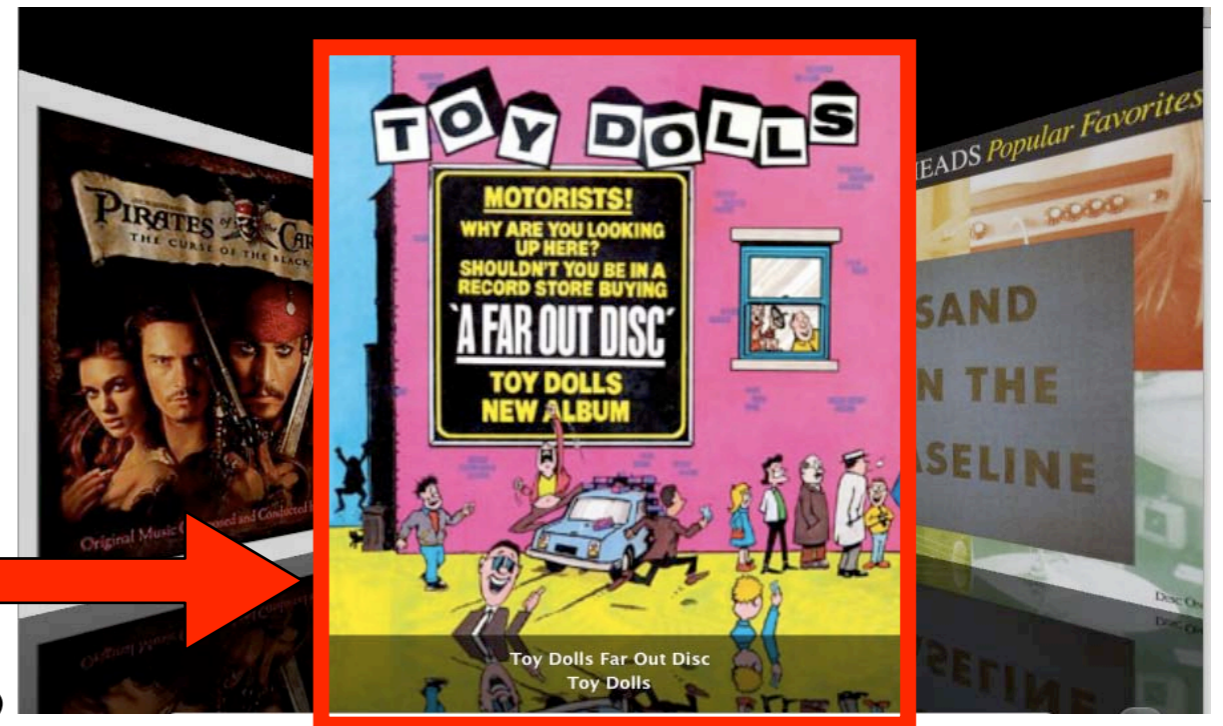
- Blending pictures together
 - **blend 1 mix two pictures together**
 - **blend 2 (from the book) overlap two pictures**
 - **blend 3 (iTunes) mirror effect**
- Homework 4

Actually, lets just overlap

- recipe 41 (page 116)
- Go try it!

The shiny floor....

- iTunes album cover
- Do this 
- Hierarchical decomposition?
- Psuedo code
 - **iTunesEffect(fileName)**
 - # get the picture, its height and create picture 50% taller picture
 - # copy the picture
 - # now put fading mirror image below picture
 - **copyPicture(source, target, startX, startY)**
 - # initialize target x and y to startX and startY
 - # for each pixel in the source, copy the pixel to the same location in the target



iTunesEffect()

■ Psuedo code (continued)

□ mirrorFade(source, target, startX, startY)

initialize target x and y to startX and startY

for each y in the target from the startY to the height of the target

figure out how much to fade to black for this row

set the row of the source (e.g. sourceY) = height of source - y to move from bottom to top of picture

for each x in the target from the startX to the width of the target

get the pixel from the source picture

multiply each color by the fade factor

put the pixel into the target

■ Notice that

□ **put x loop inside y loop to minimize # of calculations (Why?)**

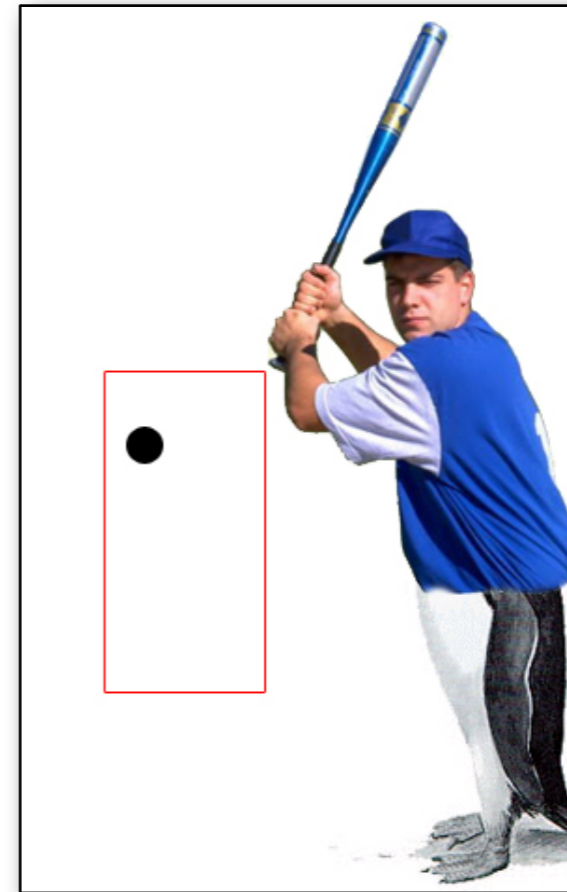
□ **x is always the same for source and target !**

Today ...

- Moving to chapter 5 ...
- HW 4
- Group project

Homework Project 4: Playoffs

- Strike or Ball?
- File with picture of ball
- ball is 20x20 pixels
- call ball or strike by printing:
 - filename **“ball”** or filename **“strike”**
- ball = outside box or touching box
- for one we'll give you the coordinates of the box
 - so **“def callBallOrStrike(file, xUL, yUL, xLL, yLL)”**
- for another you need to find the box (10 pts)
 - **“red” box (255, 0, 0) against “white” (255, 255, 255) background**
 - **a second function to call callBallOrStrike(), “def findStrikeZone(file)”**



Today ...

- Moving to chapter 5 ...
- HW 4
- Group project

Tentative group project ideas

1. Choose five pictures that represent the five elements of the world and insert 4 of them onto one which would be the background. Then posterize the entire picture.
2. Using pictures of VTech, line pictures up and fade pictures from maroon to orange.
3. Take 1 large photo, say of a face, then heavily posterize it into 4 main colors. Take the remaining 4 photos, heavily shrink them and apply 4 different color filters to them that match the 4 areas of color the large photo was posterized into. Then, sub the scaled photos into the main photo, each one acting as a larger pixel, to create the larger photo pixelized. We're basically trying to imitate this: http://www.andreaplanet.com/andreamosaic/tutorial/MosaicsWithFewTiles_files/Red_Yellow_Flower_256.jpg
4. Merge all images into a single compound image (arrangement still to be determined). Then parse through a collection of these pixels, transforming them into various sound tones (2 octaves, C5 to C7 used) based upon the luminance value of the pixel (in a similar fashion to posterization).
5. our group thought we might take pictures of each of us, and then using cropping and scaling and placement, create five collage versions of different sections of our portraits to create a five new portraits.
6. Too sketchy to report yet.
7. A picture within a picture scheme by laying pictures on top of each other to display a person looking at a screen or tv displaying people looking into screens four pictures deep.

Coming Attractions

■ Friday (9/19)

- **Assignment 3 Due 2:00 PM**

- **group project:**

- bring in your source picture
- start writing code

■ Monday (9/22)

- **Read Chapter 5**

- **Quiz 5 due 10:00 AM**

■ Friday (9/26)

- **Group project due 2:30**

- **In lab, we'll run each group abstraction on the abstraction from other groups!**

■ Monday (9/29)

- **Assignment 4 due. NOTE UNUSUAL DUE DATE!**

■ Wednesday (9/31)

- **Midterm**